TR-0863

Evaluation of the Cluster Structure on the PIM/c Parallel Inference Machine

by

T. Tarui, M. Asaie, N. Ido, T. Nakagawa
& M. Sugie (Hitachi)

Evaluation of the Cluster Structure on the PIM/c Parallel Inference Machine

Toshiaki Tarui*, Machiko Asaie*, Noriyasu Ido*,
Takayuki Nakagawa**, and Mamoru Sugie*
* Central Research Laboratory, Hitachi, Ltd.
** General Purpose Computer Division, Hitachi, Ltd.

1-280, Higashi Koigakubo, Kokubunji-shi, Tokyo 185, Japan
Tel: +81-423-23-1111
Fax: +81-423-27-7743
Email: tarui@crl.hitachi.co.jp

Abstract

The characteristics of a cluster-structure parallel computer are analyzed and evaluated on the PIM/c parallel inference machine, which includes eight-processor shared-memory clusters communicating through a communication processor connected to a network. To avoid communication bottlenecks, the maximum number of processors in a cluster is limited by the ratio of communication operations to program execution operations. On PIM/c, since this ratio is as high as 30%, the network receiving operations should be distributed to processors in the same cluster to achieve efficient eight-processor cluster operation.

Key Words

parallel computer, cluster structure, shared memory, network, communication overhead

1. Introduction

Parallel processing is the most promising technology for developing high-performance computer systems. In parallel computers with multiple processors, a mechanism to efficiently support communication between processors is one of the most important design issues. The two basic ways to connect parallel processors are with a message-passing mechanism using network communication and with a shared-memory architecture using a common bus.

With the message-passing architecture, a large-scale parallel computer can easily be developed because the network has high scalability. However, message-passing programming is not so easy as sequential programming, and communication overhead is not insignificant because packet communication which includes packet construction/deconstruction is required. For these reasons, the message-passing architecture is used mainly on large-scale parallel supercomputers with large-grain communications. With the shared-memory architecture, on the other hand, fine-grain communication between processors can be efficiently performed; however, the number of processors is limited by the common-bus throughput. The shared-memory architecture is therefore mainly used for small-scale systems with approximately ten processors.

A parallel computer, in which clusters of shared-memory multi-processors are connected through a network, has been proposed to take advantage of the efficient communication provided by shared memory [1]. This machine can be used to develop a large-scale parallel system. In this cluster architecture, the parallel processing overhead can be reduced by utilizing the communication locality of the program, in other words, by mapping groups of processes with frequent communications to the same cluster. Furthermore, hardware costs can be reduced by limiting each cluster's connection to the network to one interface, such as a communication processor. With this architecture, processors in the cluster have no network connection and thus have to communicate through the network interface. This requires inner-cluster communication through the shared memory as well as packet communication through the network interface. With a large-scale shared memory system [2], the network interface must be able to support complex operations, such as directory managements. Shared-memory multi-processor technologies, such as snooping cache, make it possible to efficiently develop a parallel computer with a cluster structure. The cluster structure is therefore one of the most important technologies needed to develop a large-scale parallel system.

The parallel inference machine (PIM) [3] was developed in Japan's Fifth Generation Computer Project driven by the Institute for New Generation Computer Technology (ICOT). The PIM was designed to efficiently execute programs written in the parallel logic-programming language KL1 [4], which allows a separate process to be defined for each step in an algorithm and which can synchronize communication between the processes. Hitachi's PIM model c (PIM/c) [5] consists of 256 processing elements (PEs) and is organized into 32 clusters of eight-processor, shared-memory multi-processors, each with a snooping cache mechanism. Each cluster has one extra processor dedicated to network communication. This processor is connected to a crossbar network. In the PIM/c, the communication processor must support the rather heavy overhead of operations involved in KL1 communication, such as address transferring, as well as the data handling.

Because the cluster structure has two levels of communication, network and common bus, conventional programs optimized for a one-level architecture (such as for a one-level, network-connected parallel computer) cannot be executed efficiently on the cluster structure. This is because communication overhead and latency is completely different between the two levels. The structure of the program must be changed to make it work efficiently on a cluster structure. The actual characteristics of a program to be run on a cluster structure have not yet been clarified in detail. Having completed the development of the PIM/c cluster system, we can now evaluate the performance of the system in detail, including the real overhead of the shared memory and network communication. We can thus investigate how parametric changes affect the system performance and determine the system architecture and programming model required to develop an efficient parallel system.

In this paper, models of programs that execute on a cluster-structure parallel computer are analyzed, focusing on network communication between clusters. The performance of application programs running on the PIM/c is measured, and the execution models are verified by the results. The characteristics and design issues of the cluster structure are also discussed. Because characteristics of the cluster structure are focused on, bare performance of network/common-bus communication is not discussed in this paper, instead the total system balance is investigated in detail.

2. The Cluster-Structure Parallel Computer

2.1. PIM/c Architecture

Figure 2.1 shows the structure of the PIM/c with a typical cluster structure. The processing elements (PEs) are connected hierarchically, nine-processor shared-memory multi-processors compose a cluster, and 32 clusters are connected through a network. Inside each cluster, eight PEs, one cluster controller (CC), and the main memory are connected to a two-way-interleaved common-bus. The 256 total PEs are contained in four cabinets. Each PE executes the KL1 programs and the CC handles network communication (it is the network interface). A snooping cache mechanism is used in the PEs and the CC to support efficient shared-memory communications.

During KL1 execution, inter-cluster communication requires heavy firmware overhead, such as address translation, real-time garbage collection, and packet construction/deconstruction. In the PIM/c, normal system firmware is designed so that all operations involved in communication are executed on the CC, in order to reduce the communication overhead on the PEs. The performance of each PE is thus increased; however, execution on the CC may become a bottleneck, even when the performance of the network hardware itself is sufficient. To reduce the CC bottleneck, the PIM/c has an additional execution mode: the network receiving operations are distributed to the PEs. The performance saturation point is thus increased, as will be discussed in Section 2.4; however, the performance of each PE decreases because PEs must execute additional communication operations.

2.2. Characteristics

In a parallel computer with a cluster structure, the PEs are connected in a two-level hierarchical architecture: between the shared memory and the network. Several PEs and a shared memory are connected by a common bus and compose a cluster of shared-memory multi-processors. The system is composed of a number of these clusters connected by a network. Each cluster also includes a network interface, such as a communication processor, connected to the network. PEs in a cluster have no network connection and thus have to communicate through the network interface. Communication between PEs in the same cluster can be performed by utilizing the high-speed shared-memory system. Communication between PEs in different clusters is done through the network, such as with packet transfer. Communication between clusters must include inner-cluster communication to the network interface. Communication overhead and latency is therefore completely different between the two levels.

Compared with a one-level, network-connected, parallel computer, a cluster structure has two significant advantages:

(1) Network hardware can be reduced to about 1/n if each cluster includes n PEs. By using low-cost shared-memory multi-processors, total system hardware costs can be reduced.

(2) Network communication can be reduced by mapping the processes which communicate frequently with each other to the same cluster. Because inner-cluster communications are performed much faster than inter-cluster communications, system performance is improved.

The cluster structure also has some disadvantages:

(3) Network throughput per PE is reduced to 1/n because the n processors share one network path. Thus, the network interface may become a bottleneck if a program has frequent communications.

(4) Inter-cluster communication latency is high, because the PEs do not have a direct network path and must use shared memory to request or receive messages through the network interface. Furthermore, if several PEs in the same cluster attempt to communicate at the same time, the requests are serialized on the network interface and message latency increases.

The following aspects must therefore be considered when designing a cluster-structure parallel system:

- performance of network communication and shared-memory communication,
- ratio of program execution and communication,
- communication locality of the program.

In the rest of this paper, we will analyze program performance in a cluster structure, focusing on communication performance and the communication-execution ratio.

2.3. Communication within a Parallel Program

Network communication within a parallel program running in a cluster structure can be classified into three types:

(1) execution dominant

When a program has few communication requirements, communication performance has little effect on system performance: execution speed increases almost linearly with the number of PEs in each cluster (provided that shared-memory multi-processor performance is efficient). This type of program is not evaluated in this paper.

(2) network latency dominant

Inter-cluster network latency occurs in a cluster structure because each PE must communicate through a network interface. When a program must wait during communication latency, it becomes idle and system performance decreases. In latency dominant programs, system performance strongly depends on network performance.

(3) network throughput dominant

If a program can hide network latency by communicating and executing programs in parallel, or when the program has a sufficient number of processes and a context-switch mechanism for communication is provided, system performance is independent of network latency. However, if network throughput becomes overloaded, system performance decreases because of communication-waiting time. Therefore, on throughput dominant programs, network throughput must be sufficient to support the communication produced by all of the PEs in a cluster.

2.4. Execution Models

In this section, we will analyze models of parallel programs executing on a cluster-structure parallel system, focusing on the network communication. The relationship between the number of PEs in a cluster and system performance is discussed in detail.

In our model, we use the following assumptions:

- Hardware throughput of the network and the common bus is sufficient.
- Shared-memory multi-processor overhead in each cluster is small enough to ignore.
- Hardware latency in the network is small compared with the overhead in the network interface and can be ignored.

Thus, the only communication overhead occurs in the network interface. On the PIM/c, because of heavy firmware overhead required in KL1 communication, these assumptions are proper as will be discussed in Section 4.1.

In our model, each PE executes those processes with the same characteristics.

We use the following notations in our analysis:

n: number of PEs in each cluster

i: system cycles used by the program to execute one process (does not include communication cycles)

c: system cycles used by the network to communicate during one process ($c < i$, that is, communication time is shorter than the program execution time)

e: system cycles used by the network receiving packets for network communication in c ($e < c$)

C: communication overhead, that is, the ratio of network communication time and program execution time (c/i)

E: receiving overhead, that is, the ratio of receiving communication time and program execution time (e/i)

Figure 2.2 shows the execution model for the throughput dominant program. In the network interface, nc cycles are used for communication during one unit of processing, because n PEs in the cluster request c cycles of communication. Since in throughput dominant programs, communication and program execution can be performed in parallel, the execution time for one process (T1) is

$$T1(n) = \max(i, nc), \qquad (1)$$

where each cluster executes n processes. Therefore, the speedup (S1) compared with the execution time with one PE is

$$\begin{aligned} S1(n) &= (n / T1(n)) / (1 / T1(1)) \\ &= n \qquad (\text{when } n < 1/C), \text{ or} \\ &= 1/C \qquad (\text{when } n > 1/C). \qquad (2) \end{aligned}$$

Linear speedup can be obtained when execution time in the network interface is less than the program execution time in the PE ($n < 1/C$). When the number of PEs is more than 1/C, the network interface becomes the bottleneck and system performance saturates.

To reduce this bottleneck, the load on the network interface must be reduced. One way to do this is to distribute portions of the communication operations to PEs in the same cluster. When this is done, the execution time for one process (T2) is

$$T2(n) = \max(i+e,\ n(c-e)), \quad (3)$$

where e cycles out of c have distributed. The speedup compared with the single-PE case (where the receiving operation is not distributed to the PEs) is

$$S2(n) = n / (1+E) \quad (\text{when } n < (1+E) / (C-E)), \text{ or}$$
$$= 1 / (C-E) \quad (\text{when } n > (1+E) / (C-E)). \quad (4)$$

By distributing the receiving operation to PEs, maximum performance and the speedup-saturation point can be increased. However, performance with a small number of PEs decreases because the workload on each PE increases by E. Distributing the receiving operations has advantages and disadvantages, so experiments on a real machine are required to determine the optimum strategy.

Figure 2.4 summarizes the performance of the above mentioned execution model, together with the latency dominant performance model discussed below.

Figure 2.3 shows the execution model for the latency dominant program. In this case, communication and program execution cannot be overlapped, they are performed sequentially. Thus, the execution time for one process (T3) is

$$T3(n) = i + nc, \quad (5)$$

where each cluster executes n processes. Therefore, the speedup compared with the single-PE execution time of a throughput dominant program is (in order to compare the speedup with the throughput dominant case, the same performance base is used)

$$S3(n) = n / (1 + nC). \quad (6)$$

As shown in figure 2.4, performance of the latency dominant program is always below the performance of the throughput dominant program, because PEs become idle during communication. The performance ratio of the two cases increases as the number of PEs in each cluster increases. At the performance saturation point of the throughput dominant program, where the difference becomes maximum, the latency dominant program shows only half the performance of the throughput dominant program.

Above discussion is based on the assumption that communication/ program-execution ratio is independent of the number of PEs in each cluster.

(In this case, the total number of program execution cycles in each cluster increases as the number of PEs increases.) In some programs, on the other hand, the total number of program execution cycles in each cluster does not increase as the number of PEs increases. In this case, the execution time for one process (T4) is

$$T4(n) = i + nc, \qquad (7)$$

where the total number of program execution cycles in each cluster is constant (i). Therefore, speedup (S4) is

$$\begin{aligned} S4(n) &= (i / T4(n)) / (i / T4(1)) \\ &= (1 + C) / (1 + nC). \qquad (8) \end{aligned}$$

S4(n) decreases as the number of PEs increases. This is because the total number of program execution cycles in each cluster is independent of n, while the communication cycles (nc) increase with n. Thus, the communication/program-execution ratio for each cluster increases as n increases.

3. Evaluation

3.1. Evaluation Method

We measured the performance of a cluster structure parallel computer using the PIM/c parallel inference machine with large-scale application programs written in the logic programming language KL1, to clarify the characteristics of the system and to investigate the parameters which determine system performance. The performance of a cluster structure is strongly affected by system balance, such as between shared-memory network performance and firmware overhead. Evaluation on a real machine is thus indispensable.

To evaluate the general communication performance of the cluster system, we used communication dominant KL1 programs (not inference dominant programs). Although KL1 is designed to solve knowledge information programs, KL1 itself is a general-purpose parallel programming language that includes the synchronization and load-balancing mechanisms essential to parallel processing.

We used two ICOT-developed KL1 programs [6] as benchmark programs:

- LSI routing program (Router)

  This throughput dominant program uses the lookahead line search method. Many routing processes are created for each grid line on the LSI and are used to execute routing in parallel. The system therefore has a large number of routing processes. The suspension/resumption mechanism of KL1 makes it possible to context-switch to another process when communication occurs, thus hiding network latency. For this reason, the Router is a throughput dominant program.

- Logic simulation program (Lsim)

  This latency dominant program is a conventional time wheel version of the parallel logic simulation program. Several (1-8) simulation engines are placed on each cluster and execute the simulation in parallel. Each simulation engine must synchronize once per simulation cycle, by sending a message to the time manager. This is done simultaneously by each PE in the cluster, so the simulation engines become idle. For this reason, Lsim is a latency dominant program.

In our experiments, we measured the relationship between the number of working PEs in a cluster and the speedup. The number of clusters was eight (maximum 64 PEs). For the throughput dominant routing program, we also measured the affect of distributing the receiving operations to PEs in the same cluster.

3.2. Results

3.2.1. Effect of the Number of PEs in a Cluster

Table 3.1 summarizes the execution time of each benchmark programs on the PIM/c with eight cluster (64 PEs). In the rest of this paper, number of clusters is fixed to eight.

Figure 3.1 shows the relationship between the number of PEs in each cluster and the speedup of the routing program. The results are consistent with the analytical results shown in figure 2.4. The execution model of the throughput dominant program discussed in Section 2.4 therefore applies to the PIM/c.

With normal execution strategy, in which all receive operations are executed on the CC, the speedup saturates at around three PEs. Above this point, the utilization of the CCs (observed using the real time performance meter) is 100% while the utilization of the PEs is only 60-80%; that is, the PEs become idle because of the communication bottleneck in the CCs. On the other hand, when the receive operations are distributed to the PEs, an almost linear speedup can be obtained. (However, performance on one or two PEs is 10-20% lower than the normal strategy.) In this case, the utilization of the PEs is almost 100% while the utilization of the CCs is around 50%. This confirms that when executing a throughput dominant program on the PIM/c, communication bottlenecks on the network interface can be solved by distributing the receive operations to the processors.

The communication overhead and other parameters discussed in Section 2.4 can be estimated from figure 3.1. The communication overhead C is 0.3 and the receiving overhead E is 0.15; that is, network communication takes as much as 30% of the program execution time and about half of the communication is spent on receiving. Thus, the performance-saturation PE number when all communications are executed on the CC is three, and eight when the receiving operations are distributed to the PEs. In the later case, however, speedup before the saturation point (when the number of PEs is one or two) is 13% lower than in the former case. The estimated parameters match well with the experimental results. (It should be noted that the value of these parameters varies with the application program.)

Figure 3.2 shows the relationship between the number of PEs in each cluster and the speedup of the simulation program. In this case, the experimental results are equal to the analytical result of the latency dominant program when the total number of program execution cycles in each cluster does not increase (discussed in Section 2.4): performance decreases as the number of PEs in each cluster increases. Speedup S4(n) is equal to the experimental result shown in figure 3.2. The communication overhead C is estimated to be 0.17, which indicates that communication overhead is smaller than for the routing program. The utilization of the PEs drops to about 30%.

In the simulation program, the communication overhead increases as the number of PEs in each cluster increases. Total execution cycles in each cluster is independent of the number of PEs because the number of the simulation gates is constant. On the other hand, communication from each cluster to the time manager increases as the number of PEs in each cluster increases, causing communication overhead to increase. Because of this, performance of simulation program decreases as the number of PEs in each cluster increases.

3.2.2. Multi-processor Overhead in the Cluster

Figure 3.1 shows that the speedup of the routing program saturates at around 3 PEs when all network operations are performed on the CC. In the performance saturation area, constant performance should be obtained according to the execution model in Section 2.4. However, the experimental results show that speedup decreases slightly when the number of PEs is more than six. This is because the shared-memory multi-processor overhead inside the cluster decreases performance.

Table 3.2 shows the number of multi-processor events per PE, measured by the hardware bus monitor, when the number of PEs in the cluster is four and eight. When the number of PEs is four, the number of cache accesses and bus-busy cycles is larger than for the eight-PE case, because each PE does twice the work. On the other hand, the lock-waiting time with eight PEs is much larger, despite the small workload on each PE. This is because, in the eight-PE case, the ratio of lock concentrations to shared resources is larger. It should be noted that common-bus throughput is sufficient, because common-bus utilization is only 5.8%. In this way, the shared-memory multi-processor overhead inside each cluster increases rapidly when the number of PEs in the cluster is close to eight because of lock concentration, which decreases system performance even when the common-bus throughput is sufficient.

When the application program has sufficient parallelism, this multi-processor overhead does not degrade performance because the speedup is more significant. However, when cluster performance saturates because of, for example, a network bottleneck, increasing the number of working PEs in the cluster beyond the saturation point degrades system performance because of the shared memory overhead. The number of PEs that can perform efficiently in each cluster is therefore limited.

4. Discussion

With a cluster structure, a large-scale parallel system can be built efficiently. However, throughput bottleneck and increasing latency in the network interface must be overcome in designing a high-performance parallel system. In this section, we will discuss the characteristics and design issues of the cluster structure parallel computer in detail.

4.1. Communication Overhead

As discussed in Section 2.4, when a communication processor is placed in each cluster to support network communication, the maximum number of PEs in one cluster is limited by the network overhead: the ratio of communication operations to program execution operations must be controlled to avoid communication bottlenecks. (The performance of the shared memory in the cluster also limits the number of PEs.) As a result, total system performance strongly depends on the communication overhead. If it is large, the number of executing PEs in the cluster should be limited to minimize the shared memory overhead inside the cluster. Furthermore, since network hardware performance is usually sufficient on a normal parallel computer, the communication overhead is primarily caused by the software involved in network communications. For this reason, communication overhead depends on the communication characteristics of the application program.

On the PIM/c, as the communication overhead for a KL1 program is as much as 30%, because of heavy firmware overhead required in KL1 communication, system performance is limited by communication bottlenecks in the CCs, the communication processors. Thus, some effort should be made to reduce the communication overhead. One approach is to distribute some of the communication operations to PEs, as will be described in the next section. The other approach is to reduce inter-cluster communication, utilizing the communication locality of the program. This is done by mapping groups of processes which communicate frequently with each other to the same cluster. In the LSI routing program, about a 20% speedup can be obtained by optimizing the process distribution.

4.2. Network Communication Execution Strategy

On a cluster-structure parallel computer with a network communication processor, system load balancing between the communication processor and the normal (program executing) processors is crucial in order to avoid communication bottlenecks. In general, communication overhead on the normal processors can be reduced and high system performance can be obtained when all communication operations are performed on the communication processors. However, this strategy may cause communication processor bottlenecks, if the throughput of the communication processors is insufficient. In this case, system performance quickly saturates.

The following strategy is therefore appropriate for cluster structures.

- When the number of normal processors in a cluster is small, or the communication overhead is not heavy, all communication operation should be performed on the communication processors.
- When the number of normal processors in a cluster is large, or the communication overhead is heavy, some of the communication operations should be distributed to the normal processors.

- Since communication overhead varies with programs, the system should be designed so that communication processors seldom become the bottleneck, because when they do, utilization of the normal processors decreases dramatically.

The communication overhead, the number of normal processors in a cluster, and the distribution of communication operations should therefore be considered when designing the network system for a cluster structure-parallel computer.

On PIM/c, because of the heavy load caused by KL1 inter-cluster communication, the communication overhead is so heavy that, when all of the communication operations are performed on the CC, system performance saturates at around only three PEs. Because of this, an improved execution strategy, in which the receiving part of the network operation is distributed to the PEs in the cluster, should be used. In this case, the maximum number of PEs in the cluster can be increased to eight. With this strategy, CC utilization is decreases to about 50% and network bottlenecks are avoided. With this strategy, however, when the number of PEs is below the saturation point (when the number of PEs is one or two), the performance of the later strategy is 10-20% lower than that of the former strategy, because the PEs must execute additional communication operations. However, since the final goal of the parallel computer is to achieve high performance with a large number of processors, the later strategy should be used, despite the poor single-processor performance.

4.3. Programming Model for the Cluster Structure

In a cluster-structure parallel computer, throughput dominant programming is needed to hide communication latency and achieve high performance. A program can be executed in a throughput dominant manner when the application itself is programmed so as to perform communication and program execution in parallel, or when the system supports a light-weight context-switching mechanism (such as the suspension/resumption mechanism in KL1) and the number of the executable processes in the application program is sufficient. With a throughput dominant program, an almost linear speedup can be obtained until the network throughput becomes the bottleneck.

On the other hand, if the program has latency dominant characteristics and the communication latency cannot be hidden, processor utilization decreases as the number of PEs in each cluster increases, and the performance degradation ratio compared with the throughput dominant program increases. This is because in a cluster-structure parallel computer, communication requests from PEs in a cluster are serialized on the CC and communication latency increases with the number of PEs.

Furthermore, if the program has no communication locality, so that communication overhead increases with the number of PEs, system performance decreases as the number of PEs increases. The logic simulation program evaluated in this paper is one example of this. In this case, only one PE in the cluster should be executed, even if the cluster has several PEs.

To avoid the performance degradation in a latency dominant program, the program algorithm should be completely changed to a throughput dominant algorithm. For example, a time warp algorithm suitable for parallel execution has been proposed [6] for the logic simulation program. In this algorithm, synchronization through the time manager is not used so that parallel execution of simulation and communication is possible.

5. Conclusion

In this paper, we analyzed and measured the performance of a cluster-structure parallel computer on the PIM/c parallel inference machine, which includes eight-processor shared-memory clusters, each connected to the network with a communication processor, to clarify the characteristics of the cluster structure.

The main results obtained from our investigation are the following.

(1) In a cluster-structure parallel computer, the communication bottlenecks in the network interface should be avoided. Therefore, the

maximum number of PEs in each cluster is limited by the communication overhead (the ratio of operations required for communication to those required for program execution), in addition to the shared memory overhead in each cluster. As a result, total system performance depends strongly on the communication overhead. System performance saturates when the number of processors in the cluster exceeds the limitation.

On PIM/c, since the communication overhead for KL1 inference operation is as much as 10-30%, the number of PEs in a cluster is limited to three if the communication processor (CC) performs all of the communication operations. Therefore, the communication distribution discussed next is indispensable for eight-PE clusters.

(2) To decrease bottlenecks in the communication processors, some communication operations should be distributed to program-executing processors in the same cluster. With this inner-cluster load balancing, communication overhead can be reduced and the maximum number of processors in a cluster can be increased.

On PIM/c, the workload on the CC can be sufficiently reduced, by distributing the network receiving operations to the PEs so that the maximum number of PEs in a cluster can be increased to eight.

(3) The programming algorithm on a parallel computer should be designed so as to hide communication latency by performing communication and program execution in parallel. If a processor has to wait due to communication latency, processor utilization decreases. In a cluster structure parallel computer, latency hiding is indispensable because the communication requests from processors in the cluster are serialized in the network interface, increasing communication latency.

(4) In latency dominant programs, increasing the number of processors in the cluster without consideration of communication locality may result in decreased system performance because of increasing communication overhead. In this case, executing one processor in each cluster gives the best performance, even if the cluster has several processors. To solve this problem, application programs should be changed to reduce inter-cluster

network communication by utilizing communication locality. Communication overhead should not increase as the number of PEs in a cluster increases.

Acknowledgements

The authors would like to thank Dr. Shun'ichi Uchida, Manager of the Research Department of ICOT, for his guidance and support, and Dr. Takashi Chikayama, Chief of the 1st ICOT Laboratory, for his helpful discussions. This research was sponsored by the ICOT.

References

[1] N. Hamanaka, J. Nakagoshi, and T. Tanaka, "Reducing Network Hardware Quantity by Employing Multi-Processor Cluster Structure in Distributed Memory Parallel Processors," Parallel Processing: CONPAR 92 - VAPP V, Springer-Verlag 634, pp. 25-30, 1992.

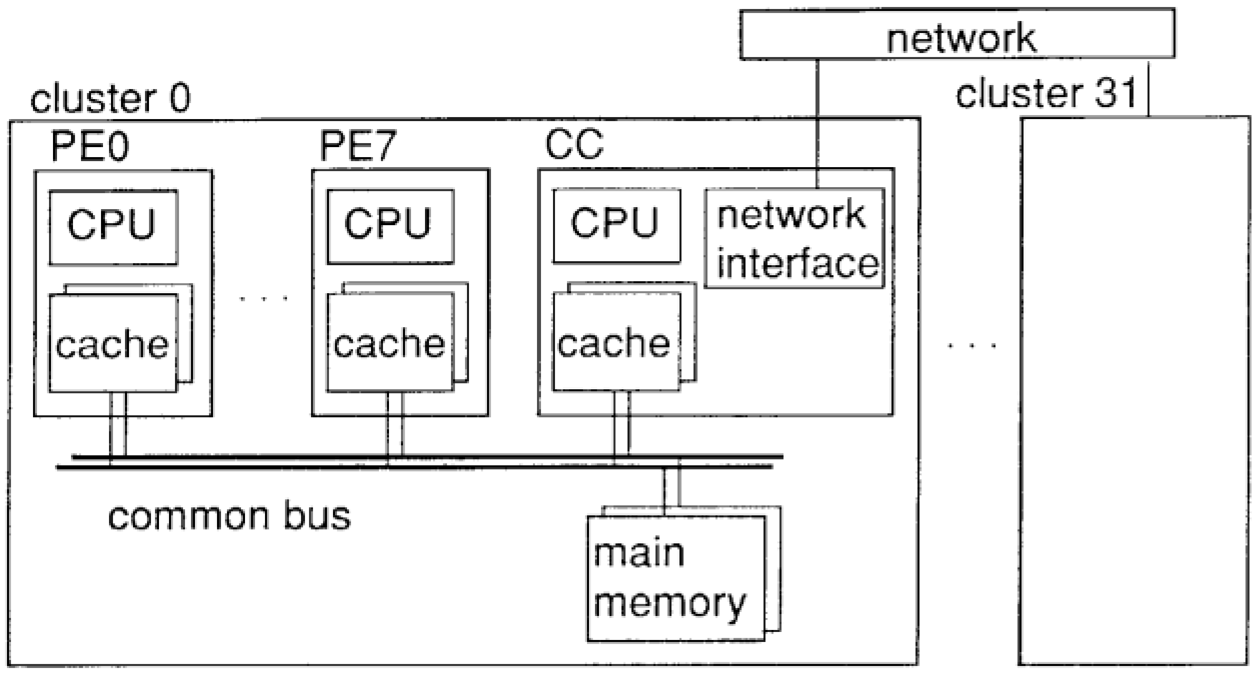
[2] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "The Stanford Dash Multiprocessor," IEEE Computer, Vol. 25, No. 3, pp. 63-79, 1992.

[3] S. Uchida, "Summary of the Parallel Inference Machine and its Basic Software," Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp. 33-49, 1992.

[4] T. Chikayama, "Operating System PIMOS and Kernel Language KL1," Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp. 73-88, 1992.

[5] T. Nakagawa, N. Ido, T. Tarui, M. Asaie, and M. Sugie, "Hardware Implementation of Dynamic Load Balancing in the Parallel Inference Machine PIM/c," Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp. 723-730, 1992.

[6] H. Date, Y. Matsumoto, K. Kimura, K. Taki, H. Kato, and M. Hoshi, "LSI-CAD Programs on Parallel Inference Machine," Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp. 237-247, 1992.



PE: processing element
CC: cluster controller

Fig. 2.1. PIM/c structure.
(PIM/c: Parallel Inference Machine/Model C)

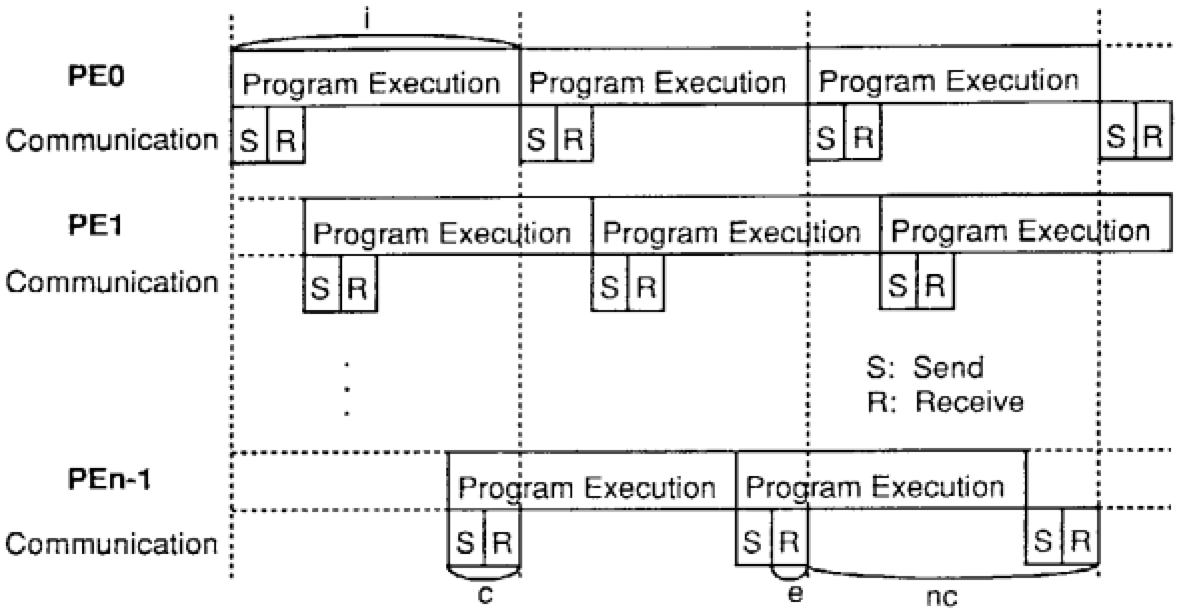


Fig. 2.2. Execution model for throughput dominant program.

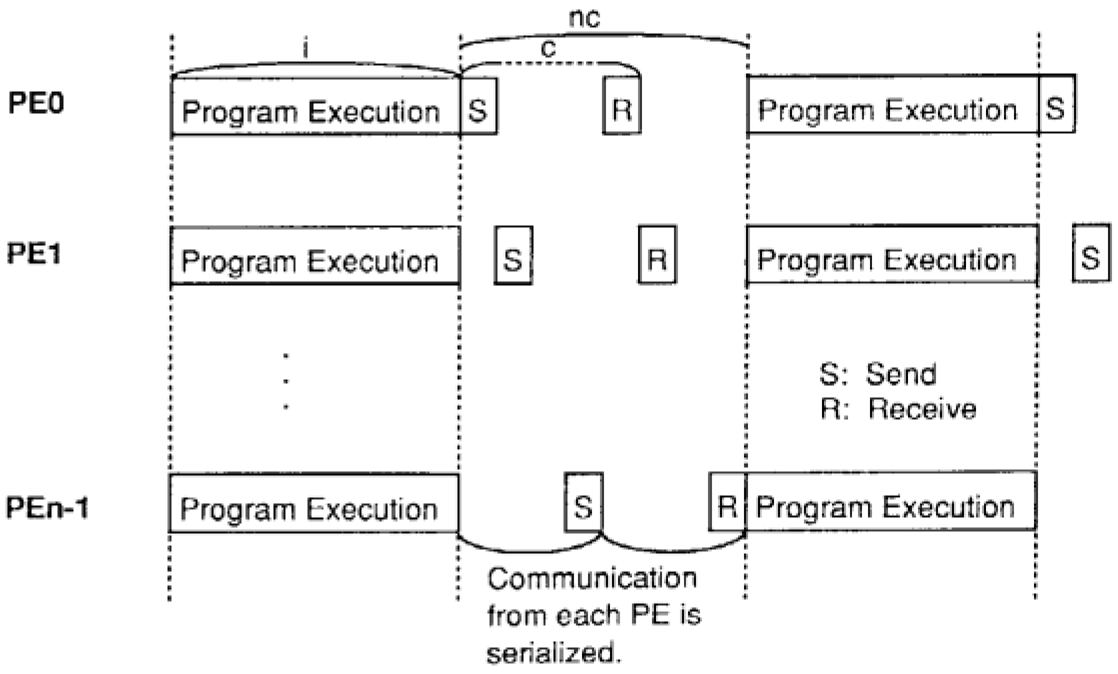


Fig. 2.3. Execution model for latency dominant program.

C: ratio of network communication time
and program execution time (communication overhead)
E: ratio of receiving communication time
and program execution time

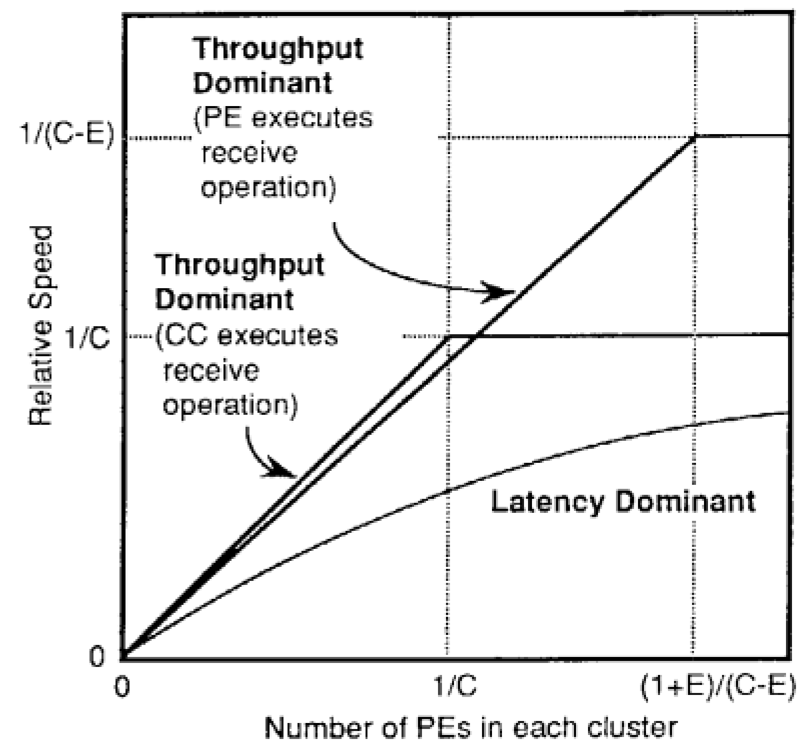


Fig. 2.4. Throughput dominant and latency dominant program performance.

PE: processing element
CC: cluster controller

Table 3.1. Execution Time of Benchmark Programs

Benchmark Program	Routing Program		Simulation Program
	CC executes receive operations	PE executes receive operations	
Execution Time (seconds)	53.4	22.5	190.0

8 clusters (64 PEs)

Table 3.2. Number of Multi-Processor Events per PE

(thousand times) (thousand cycles)

Number of PEs in a Cluster	Number of Cache Accesses	Number of Bus Accesses	Bus-Busy Cycle Time	Lock-Waiting Cycle Time
4	47,559	549	4,184	406
8	31,594	554	3,458	3,114

Routing program.
CC executes receive operations.

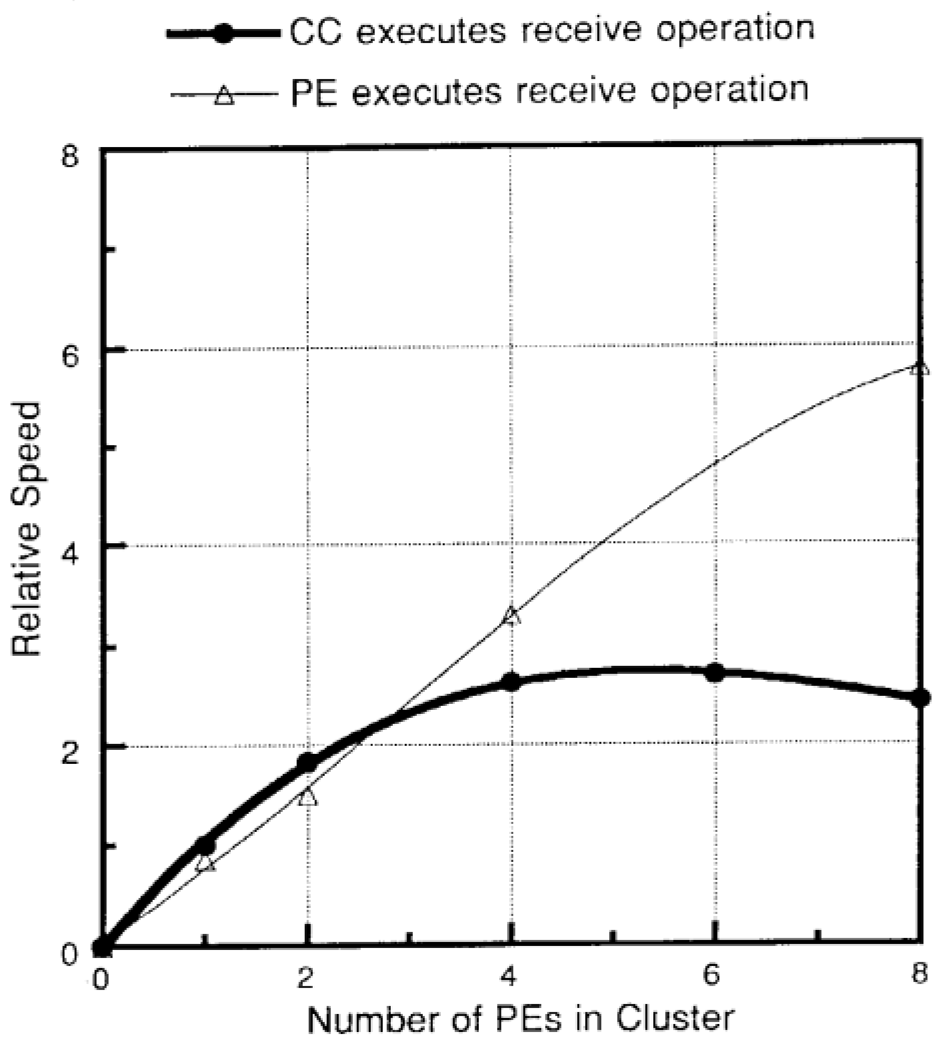


Fig. 3.1. Relationship between number of PEs and speedup of routing program.

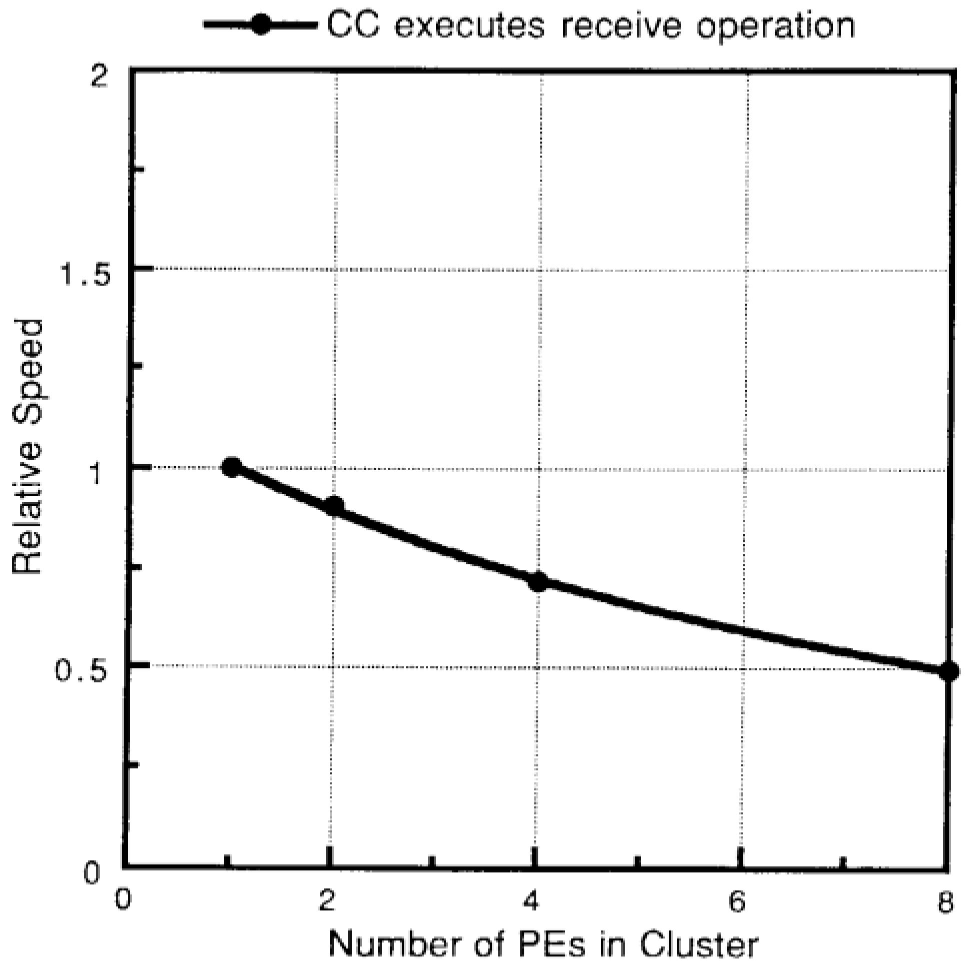


Fig. 3.2. Relationship between number of PEs and speedup of logic simulation program.